Sistemas de E/S

Indice

1. Problemática de la E/S

A pesar de que los periféricos son imprescindibles para comunicar la máquina con el exterior, la CPU no puede “dialogar” directamente con ellos.

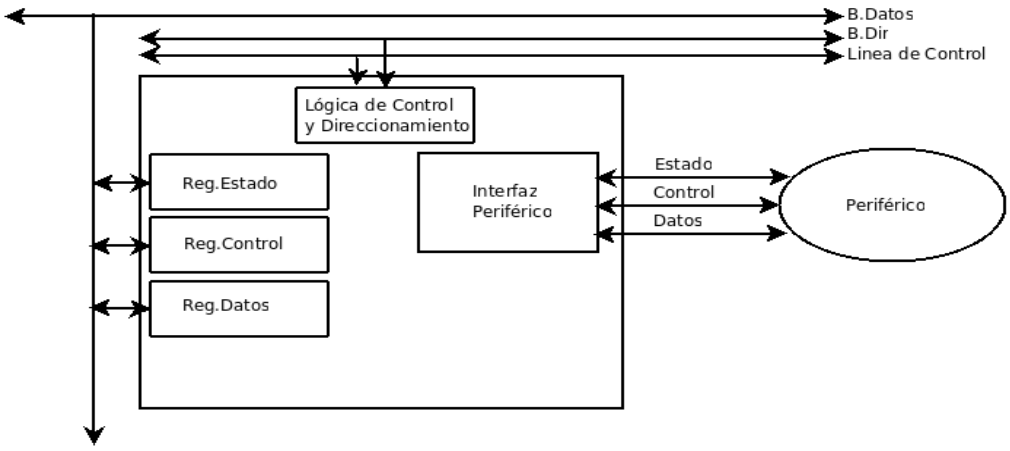
Por tanto surge la necesidad de integrar un módulo de E/S cuya función es ocultar las particularidades de cada periférico. Dichas particularidades pueden ser la velocidad de transferencia, modo de funcionamiento, formato o tamaño de los datos.

2. Módulos E/S

- *Funciones*:

  1. Control y temporización de la transferencia de datos
  2. Comunicación con la CPU
  3. Comunicación con el periférico
  4. Almacenamiento temporal de datos (buffering)
  5. Detección de errores

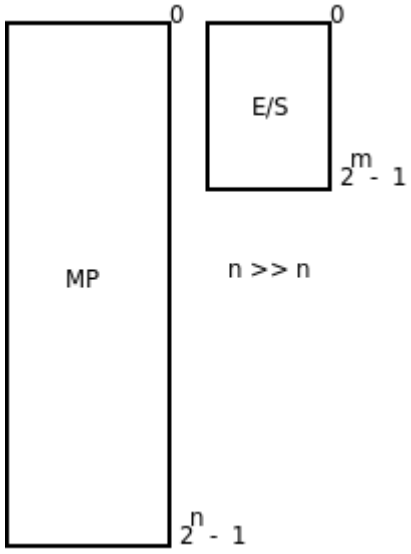
- *Estructura general*:



3. Instrucciones de E/S

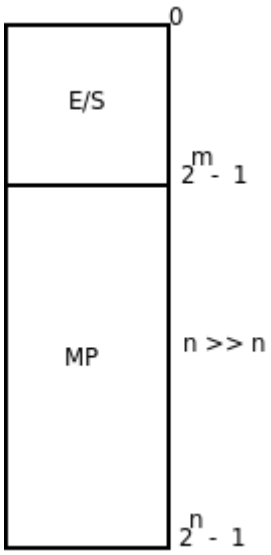
```
IN .RD/DIR(Perif)        ; CPU ← Mód E/S
OUT .RD/DIR(Perif)       ; CPU → Mód E/S
```

- *Mapa de E/S Separado*



Tenemos las mismas direcciones, por tanto debemos utilizar instrucciones específicas y añadir una señal $\overline{IOREQ}$ frente a $\overline{MEMRQ}$.

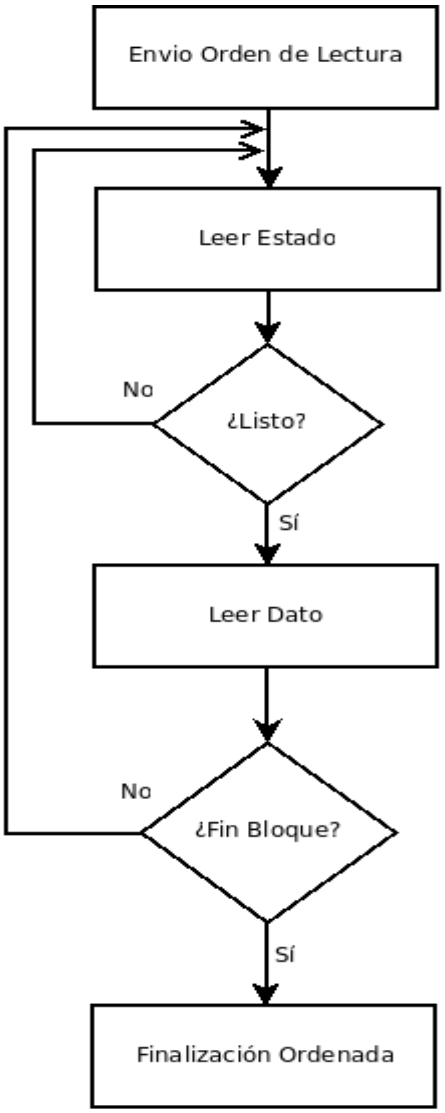
- *Mapa de E/S Común*



No hacen falta instrucciones adicionales ni la señal $\overline{IOREQ}$. La distinción de una operación de E/S con otra de Memoria se realiza mediante la dirección.

4. Técnicas de E/S

Las diferentes técnicas de E/S vienen dadas por la necesidad de aprovechar mejor la CPU.



¿Cómo afecta a las distintas técnicas?

Trabajo CPU/Tipo	I	S	T	F
Programada	X	X	X	X
Interrupcion	X		X	X
DMA	X			X

X = Desperdicio de CPU

4.1 E/S Programada

Todas las fases son controladas por la CPU mediante la ejecución de instrucciones. Se utiliza para dispositivos pequeños.

R.Control	{	xxx...0 → OFF		xxx...1 → ON	}
R.Estado	{	xxx...0 → Busy		xxx...1 → Ready	}

```
I:  ld .R2, #8              ; Contador = 8
    ld .R3, /DirMem         ; Dirección de Destino
    ld .R1, /R_Control_P1   ; Encendido P1

S:  in .R1/R_Estado_P1      ; Estado P1
    cmp .R1, #1             ; ¿Está Listo?
    bz $S

T:  in .R1/R_Datos_P1
    st .R1,[.R3]
    inc .R3                 ; Siguiente posición de memoria
    dec .R2                 ; Decrementar contador
    bnz $S

F:  ld.R1, #0               ; Orden OFF
    out .R1/R_Control_P1    ; Apagado P1
```

4.2 E/S por Interrupciones

Aquí liberamos a la CPU de la fase de sincronización. El módulo produce interrupciones cuando el dato está listo.

La CPU posteriormente realiza una instrucción de transferencia.

Utilizando la señal $\overline{INT}$ vemos como se evita el bucle infinito durante FETCH.

Antes:		Después:	
fetch:	IR ← M(PC) PC ← PC +1	N_fetch:	SI $\overline{INT} \wedge \overline{SRI}$ CRI SINO bsr fetch
CRI:	PUSH(PC) PUSH(SR) $SR.\bar{I}=1$ $SR.S=1$ Calcular dirección de RTI …..... jump RTI	RTI:	POP(SR) POP(PC) Transferencia

Apéndice: Varios periféricos

i) Identificación y Localización RTI
ii) Prioridades
iii) Anidamiento RTI

i) Identificación y Localización RTI

Software (polling): se pregunta dentro de la RTI a los módulos E/S cuál es el dispositivo que solicita la interrupción. **No hay anidamiento.**

Hardware (Vectorización): se incorpora una nueva señal $\overline{INTA}$ Reconocimiento de instrucción que la CPU activa para notificar al módulo de E/S que reconoce y atiende la interrupción.

El módulo deposita el identificador (vector) en el bus de datos. La CPU lo lee y sabe quién solicitó la interrupción.

El CRI se modifica de la siguiente forma:

CRI: PUSH(PC)
PUSH(SR)

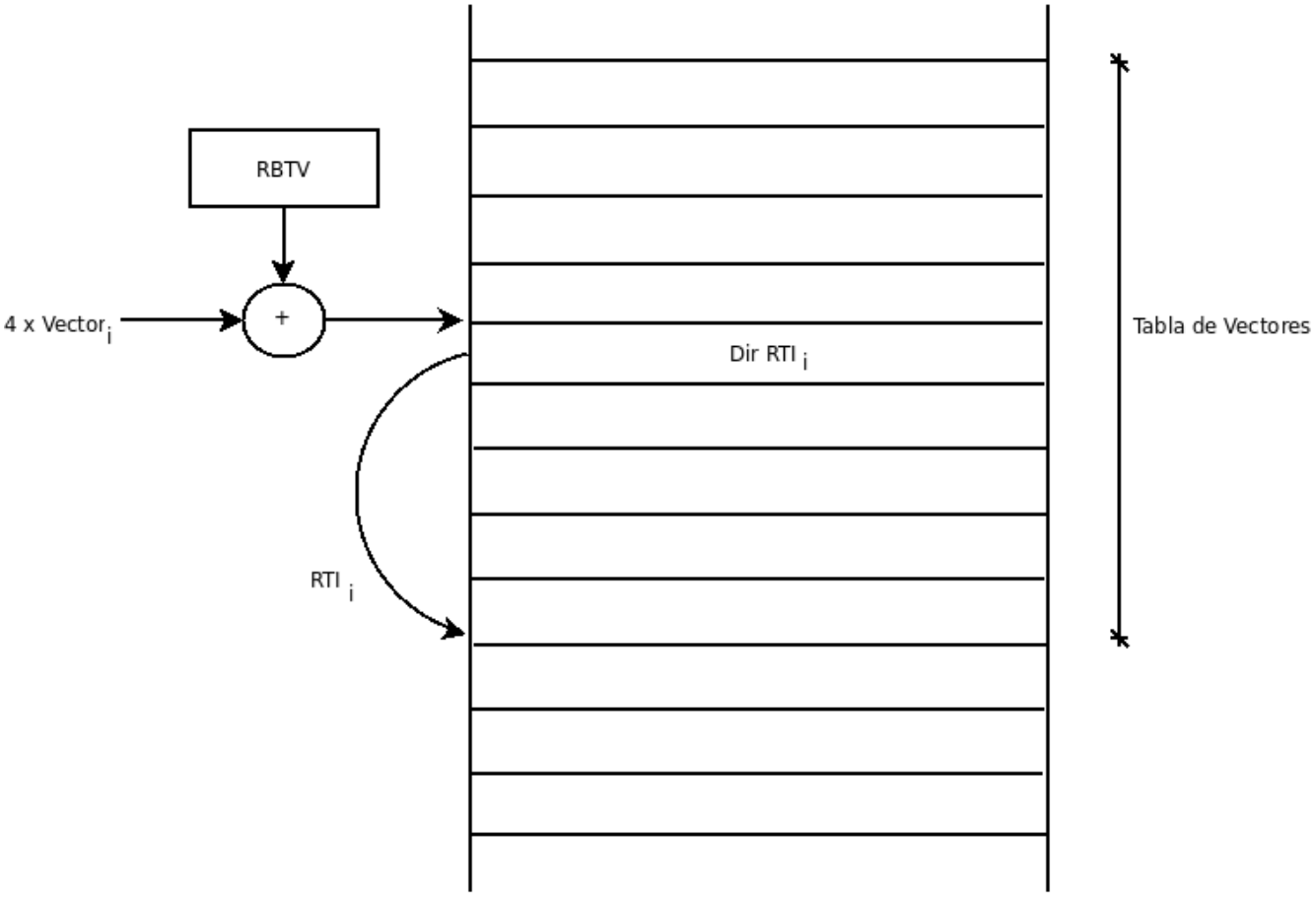
$SR.\bar{I}=1$
$SR.S=1$

Ciclo de Bus de Reconocimiento $\overline{INT}$

Activar $\overline{INTA}$
Leer vector I de Bus de Datos
Desactivar $\overline{INTA}$

PC ← b (vector I)

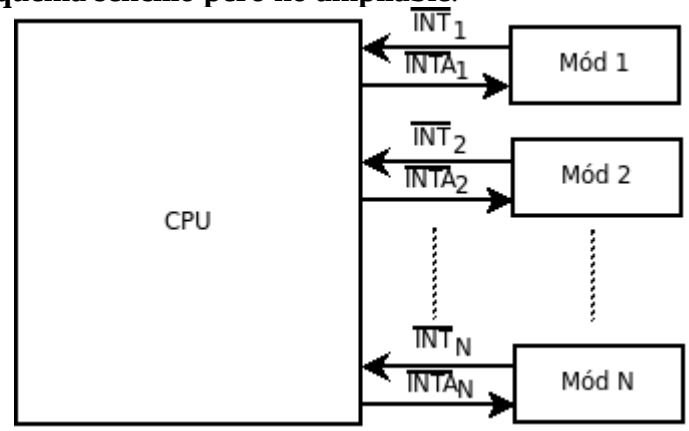
En el caso del Motorola 6800 se hace con Direccionamiento Relativo a Reg.Base, que es el de la Tabla de Vectores, también conocido como RBTV.



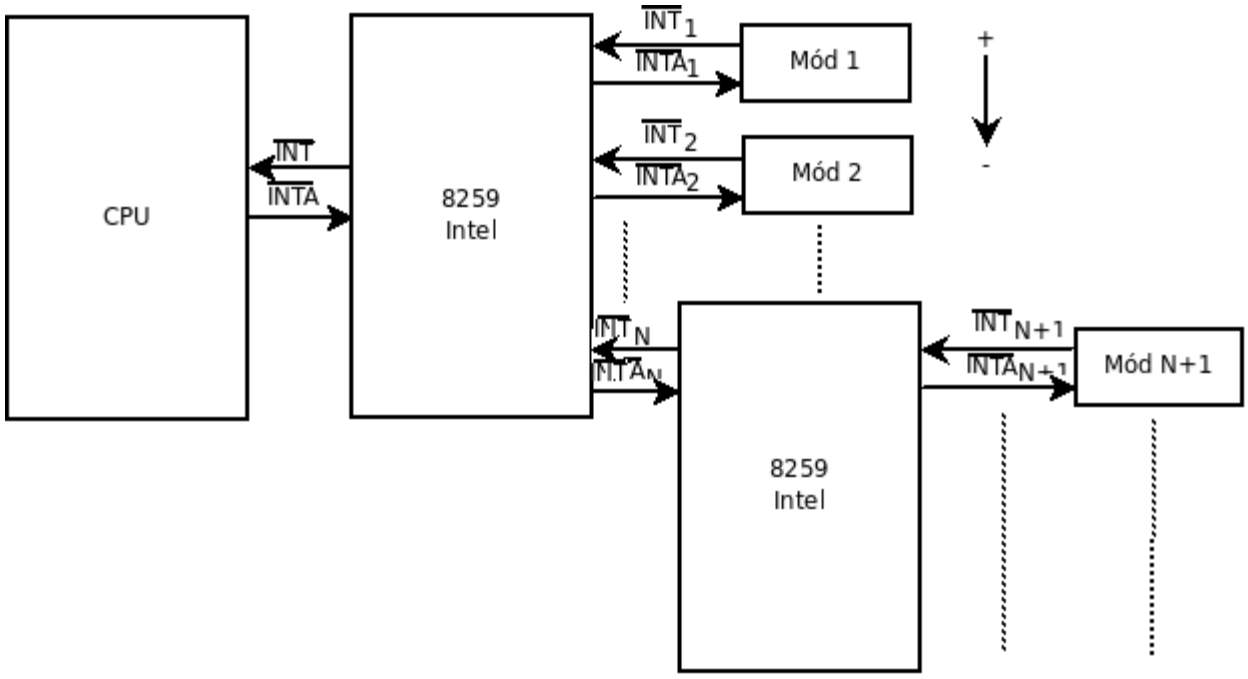
ii) Prioridades

Disponemos de tres tipos de esquemas para establecer la prioridad a nivel HW:

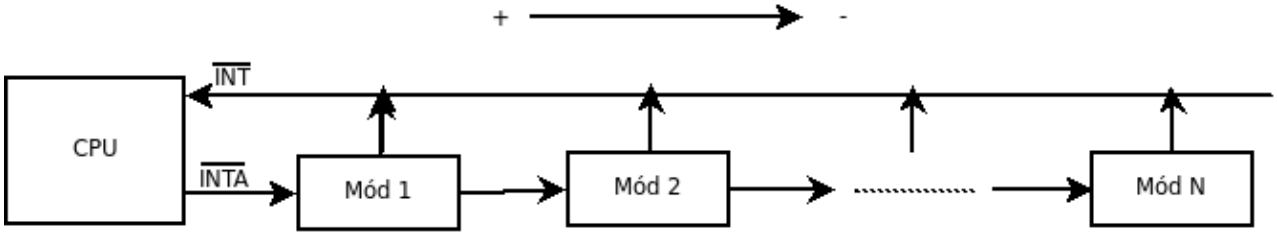
- *Centralizado:* **esquema sencillo pero no ampliable**.



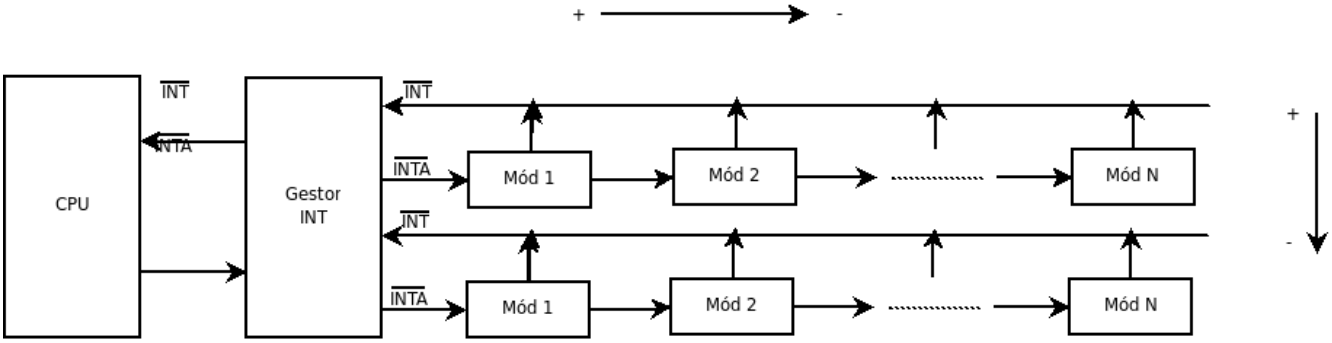
En la vida real (ejemplo Intel 8259) se añade un circuito, un gestor de interrupciones entre CPU y módulos.



- *Encadenado:* **esquema poco flexible y ampliable**. Posee la limitación de que la señal de $\overline{INTA}$ debe propagarse con suficiente rapidez para que la CPU no lea basura del Bus de Datos.

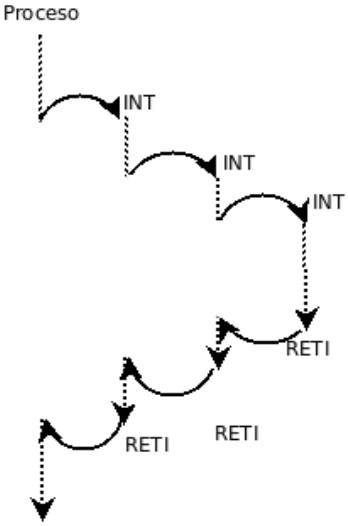


- *Híbrido:* mezcla ambos esquemas para conseguir **mayor amplitud y flexibilidad.**



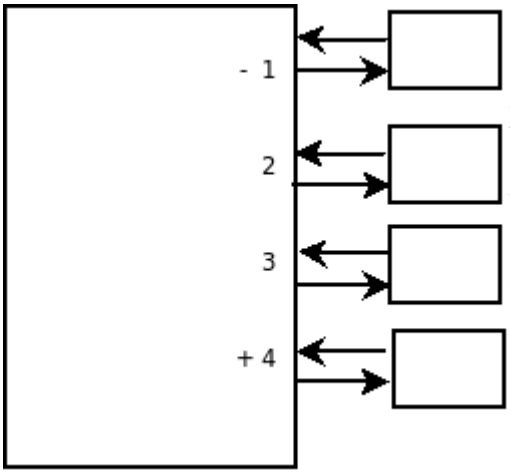
iii) Encandenamientos RTI

Se realizarán preferentemente con el esquema centralizado.



Este diseño es inviable con el esquema encando puesto que no conocemos la prioridad del dispositivo que ha solicitado la interrupción.

Ejemplo 6800:



En vez de disponer de un bit en el SR para almacenar el estado de las interrupciones (habilitadas o inhibidas) tendremos 3 bits para guardar el nivel de interrupción que está siendo atendido. Si llega una solicitud con mayor prioridad se atiende. El 0 indica que no existe interrupción. El nivel 7 siempre se atiende, es no enmascarable.

Estos tres bits son la máscara de interrupción (MI).

Nuevo CRI: SI [(I == max $\overline{INT_j} \forall j$) ^ (I > SR.MI v I==7)]
PUSH(PC)
PSUH(SR)
SR.MI = I ; Inhibimos las interrupciones del nivel ≤ I
SR.S = 1 ; Modo supervisor

Ciclo de Bus de Reconocimiento $\overline{INT}$

Activa $\overline{INTA_i}$
Leer Vector del Bus de Datos
Desactiva $\overline{INTA_i}$
PC ← DirRTI en funcion del Vector

Obs *Asignación de Prioridades*

a) Según la "importancia" del perférico (alarmas, fallos de sistema...)
b) Según el tiempo entre interrupciones. A mayor frecuencia de interrupciones tendrá mayor prioridad.

4.3 E/S por DMA

Además de la sincronización el mód E/S se encarga ahora de la transferencia. Sólo la CPU se encarga del Inicio y Fin.

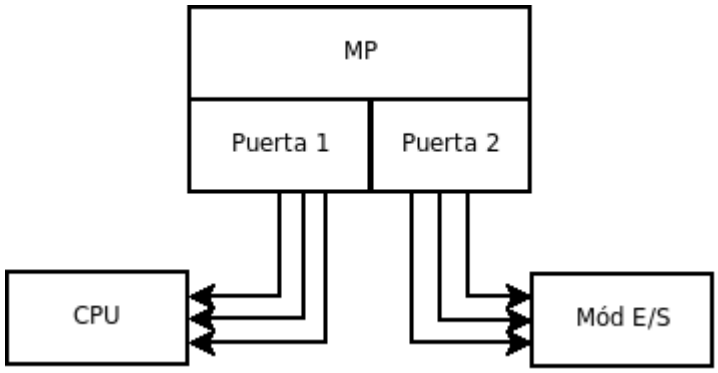
El módulo va dejando los datos en Memoria Principal y esto aumenta su complejidad. Debe generar las señales de control necesarias (RD,WR,MEMRQ) y debe acceder a los Buses de Dirección y Datos.

Además debe conocer la dirección donde deja los datos y el contador de datos así como tenerlos centralizados (dir++, cont--).

Se añaden dos registros más al módulo de E/S: uno para la Dirección de Memoria Principal y otro para el contador, así como para actualizarlos.

Implementaciones

1.) Mp Multipuerta

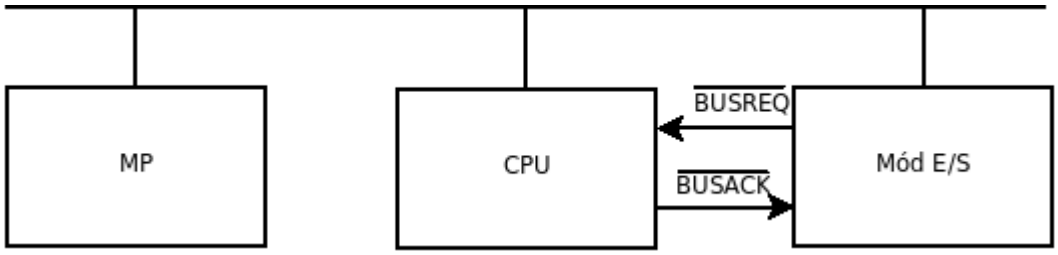


Se permite el acceso simultáneo a Mp desde distintas puertas. La Mp bloquea y evita colisiones en la misma posición.

Esta memoria es más compleja y costosa. Poco flexible, tiene un nº limitado de puertas.

En la actualidad **no se utiliza**.

2.) Mp con una puerta y buses compartidos



Se añaden dos nuevas señales para “dialogar” entre CPU y Mód sobre el uso de los buses. Las señales $\overline{\text{BUSREQ}}$ y $\overline{\text{BUSACK}}$ significan solicitud y concesión respectivamente. La CPU puede conceder los buses a mitad de una instrucción. Una vez concedidos la CPU no puede usarlos, se debe parar (pérdida de tiempo).

Cuando el módulo de E/S tiene el buffer lleno (lectura) o vacío (escritura) necesita acceder a Mp para transferir datos. Solicita los buses a la CPU con $\overline{\text{BUSREQ}}$, cuando la CPU puede los concede mediante la señal $\overline{\text{BUSACK}}$. Al ver esta señal el Mód E/S realiza la transferencia y la CPU entonces desactiva $\overline{\text{BUSACK}}$.

Distinguimos pues dos funcionamientos:

- Modo aislado: se transfieren los datos de uno en uno.
- Modo ráfaga: transfiere n datos realizando n accesos a Mp.

La operación de E/S finaliza cuando el contador del módulo llega a cero. En ese momento mediante una interrupción se avisa a la CPU de su finalización.

La CPU ejecutará entonces la RTI (que no realiza transferencia) y se limita a finalizar la operación de E/S.